UN ENTORNO PARA EL APRENDIZAJE DE LA PROGRAMACIÓN

Norma Moroni - Perla Señas

Grupo InE
Departamento de Ciencias de la Computación
Instituto de Ciencias e Ingeniería de Computación
Universidad Nacional del Sur.
Av. Alem 1253 - 8000 - Bahía Blanca - ARGENTINA
moroni@criba.edu.ar ccsenas@criba.edu.ar

Resumen

Un curso introductorio de programación puede diseñarse siguiendo distintos paradigmas. Se advierte una búsqueda permanente de nuevas estrategias metodológicas que permitan abordar la tarea con mayor probabilidad de éxito, cualquiera sea el modelo elegido. La computadora que es propuesta como herramienta para la enseñanza de otras disciplinas debiera ser la herramienta natural para la enseñanza de la programación en todas sus etapas. Si partimos de la necesidad de desarrollar un curso introductorio basado en el paradigma imperativo y siguiendo una metodología estructurada y modular, es importante disponer de un lenguaje algorítmico menos rígido que un lenguaje de programación. Se propone un entorno de programación adecuado que permite al alumno desarrollar los algoritmos trabajando directamente sobre la computadora evitando tener que recordar expresamente detalles del diseño estructural de los mismos.

UN ENTORNO PARA EL APRENDIZAJE DE LA PROGRAMACIÓN

1. Introducción

Un curso introductorio de programación puede diseñarse siguiendo distintos paradigmas. Existen justificaciones y experiencias realizadas que avalan cada una de las posturas. Se advierte una búsqueda permanente de nuevas estrategias metodológicas que permitan abordar la tarea con mayor probabilidad de éxito, cualquiera sea el modelo elegido. En algunos casos se ha presentado una máquina hipotética capacitada para realizar sólo algunas operaciones primitivas, en otros, simulaciones de las tareas de un robot circulando por una ciudad o recorriendo un laberinto [Peyrin],[Pyott].

Debe pensarse que esta forma de introducir el aprendizaje de la programación requiere posteriormente una adaptación, que no siempre es natural, para continuar con cursos de programación más avanzados. Ante esto surge la cuestión: por qué no introducir a los alumnos en el medio ambiente de la máquina con la que van a seguir trabajado posteriormente?. Debiera ser natural que la computadora que es propuesta como herramienta para la enseñanza de otras disciplinas se constituya también en herramienta útil para la enseñanza de la programación en todas sus etapas [Chavey]. Los seguidores de la POO aseguran que dicho paradigma ofrece esa posibilidad. No obstante no debe olvidarse que los conceptos de programación estructurada están presentes en el modelo OO [Wu].

Si partimos de la necesidad de desarrollar un curso introductorio basado en el paradigma imperativo y siguiendo una metodología estructurada y modular, debemos considerar diferentes aspectos:

a) Cómo expresar con precisión el problema a resolver?

b) Qué estrategia de resolución elegir?

c) Cómo plasmar la resolución en un algoritmo?

d) Cómo escribir el programa correspondiente en un lenguaje de programación?

Es importante que el alumno desarrolle capacidades que le permitan encarar la resolución de un problema de manera eficaz. Por otra parte, la metodología empleada en este aprendizaje debe tender a que pueda resolver problemas cada vez más complejos. Adquirir habilidades para escribir algoritmos no es una tarea trivial. Sería conveniente que los algoritmos se expresaran en lenguaje natural y que se contara con herramientas adecuadas que facilitaran su elaboración.[Geitz]

En esta búsqueda de recursos metodológicos se ha pensado muchas veces en la importancia de disponer de un lenguaje algorítmico (menos rígido que un lenguaje de programación). Esta propuesta presenta ventajas tales como:

- Centralizar la atención en la resolución del problema y no en los detalles propios de la rigidez de un lenguaje de programación.

- Atenuar los inconvenientes que surgen en un curso integrado por alumnos con distintos niveles en el manejo del lenguaje de programación.

- Poder adquirir habilidades y conocimientos importantes para el desarrollo de tareas de programación, que podrá aplicar independientemente del lenguaje con el que finalmente codifique.

Se han advertido, sin embargo, algunos problemas en su aplicación:

Para un alto porcentaje de los alumnos, la elaboración de algoritmos "con lápiz y papel" resulta una tarea pesada y poco atractiva. Muchos de ellos aducen que el desarrollo de esa tarea no cubre sus expectativas, ya que lo que desean es "trabajar con la computadora". Esto ha llegado a plantearse de manera tan fuerte en algunos casos que algunos docentes han tratado de paliar la situación proponiendo el uso de un procesador de texto para la escritura de los algoritmos.

Se ha comprobado también una marcada tendencia en los alumnos a abandonar el lenguaje algorítmico una vez que aprenden el lenguaje de programación. Esto si bien les permite usar directamente la computadora, crea hábitos poco recomendables si se tiene presente que futuros desarrollos serán más grandes y más complejos.

2. Sobre el entorno

2.1. Justificación

Se ha pensado que una buena solución para salvar los problemas planteados sería contar con un entorno adecuado que permitiese al alumno desarrollar los algoritmos trabajando directamente sobre la computadora y sin necesidad de tener que recordar expresamente detalles del diseño estructural de los mismos.

2.2. Descripción

Este entorno contará con:

a) Un editor interactivo de algoritmos que ofrecerá aquellos elementos necesarios para una programación estructurada y modular. De esta manera el alumno podrá centrar su atención en cómo plantear el algoritmo, despreocupándose de los detalles de la escritura.

b) Un constructor automático de trazas que permitirá visualizar la secuencia de estados de la computación.

c) Un traductor que brindará la posibilidad de obtener el programa[1] correspondiente a un algoritmo (y subalgoritmos involucrados) que se ha editado previamente. De esta manera el

[1] Se ha pensado en este caso en lenguaje Pascal, aunque podría resolverse para otro lenguaje de programación.

alumno podrá obtener automáticamente el programa que él mismo debería escribir en una próxima etapa de codificación. La posibilidad de contar con este traductor lo beneficiará en:

- Poder captar características del lenguaje de programación por observación directa de programas, en comparación con los algoritmos que ha confeccionado previamente y que han dado lugar a dichos programas. Esto plantea un acercamiento al lenguaje de programación similar al que se realiza en el aprendizaje de la lecto-escritura de un lenguaje natural. Sabido es que en el caso de los lenguajes naturales, los aprendizajes de lectura y escritura se desarrollan en forma paralela y coordinada, basándose uno en el otro.

- Poder compilar y ejecutar el programa obtenido en la traducción.

- Poder comprobar los resultados de los algoritmos para distintas muestras de entrada sin necesidad de conocer el lenguaje de programación.

- Poder diferenciar conceptos tales como: algoritmo, programa, ejecución de un programa, lenguaje algorítmico, lenguaje de programación, entidad estática, entidad dinámica, entre otros.

2.2.1 El editor de algoritmos

El trabajo con el editor de algoritmos será de tipo interactivo. Ofrecerá al alumno las estructuras básicas de un lenguaje algorítmico que él completará según cada caso. Cada una de esas estructuras constará de *textos fijos* y de *textos reemplazables*. Los textos fijos quedarán en el algoritmo, y los reemplazables deberán ser sustituidos de acuerdo a su semántica. Por ejemplo la estructura de control condicional será ofrecida por el editor de la siguiente manera:

Si *condición* **Entonces** *acción*[2]

Aquí **Si** y **Entonces** constituyen los textos fijos, mientras que *condición* y *acción* son los textos reemplazables que deberán ser sustituidos por una expresión booleana y una estructura de control, respectivamente. El editor ofrecerá facilidades para la creación de los textos de reemplazo, controlará su validez, y en caso de detectar entradas no válidas ofrecerá mensajes orientadores y la posibilidad de realizar la corrección.

La interface del editor usará recursos gráficos con el fin de lograr una mejor visualización y comprensión de los algoritmos. Brindará la posibilidad de ver al mismo tiempo varios algoritmos, usará distintos colores para distinguir los textos fijos de los de reemplazo, proporcionará cajas extensibles para agrupar las acciones de una secuencia, entre otras facilidades.

El editor pondrá a disposición del usuario las siguientes opciones:

- Estructura general de un algoritmo.

[2] En adelante se indicarán en negrita los textos fijos y en itálica los textos de reemplazo.

- Asignación.

- Estructuras de control: secuencia de acciones, condicionales, repeticiones, invocación a un algoritmo.

- Conjuntos de valores que puede tomar un dato determinado, entre ellos: conjunto de enteros, conjunto de caracteres, conjunto booleano, conjunto de sucesiones finitas homogéneas.

Al seleccionar una opción, la respuesta del editor consistirá en lo siguiente:

- Si la opción es **Estructura general de un algoritmo**, se visualizará:

Algoritmo *nombre*
Datos de Entrada: *lista de nombres*
Datos de Salida: *lista de nombres*
secuencia de acciones
Fin

Observación: Una vez ingresadas las *listas de nombres* de los Datos de Entrada y de los Datos de Salida, el editor exhibirá la lista de dichos nombres a los cuales se les deberá asociar el conjunto de valores correspondiente a cada uno de ellos, por ejemplo:

Datos de Entrada: x , y

Datos de Salida: z

x $\in$ *{ valores }*
y $\in$ *{ valores }*
z $\in$ *{ valores }*

Luego se sustituirán en cada caso los textos reemplazables *{ valores }* por los correspondientes conjuntos.

- Si la opción es una **asignación** ($\leftarrow$), se visualizará:

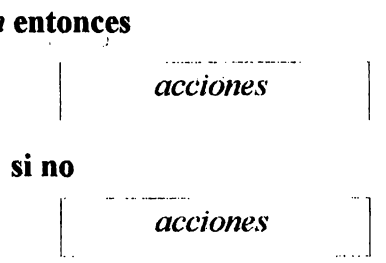
nombre $\leftarrow$ *expresión*

- Si es una **secuencia de acciones,** se visualizará:

acciones

Al reemplazar *acciones* por la sucesión de estructuras de control, la caja rectangular que las encierra ajustará su tamaño automáticamente.

- Si se trata de una estructura condicional, se visualizará:



Se especificarán luego los textos reemplazables, y se podrá cancelar la opción **si no.**

En forma análoga se resuelve para las restantes estructuras de control.

2.2.2 El traductor

Al activarse el traductor, se solicitará el nombre del algoritmo a traducir. Luego aparecerá en pantalla el programa correspondiente en lenguaje Pascal. Si el algoritmo que se traduce tiene llamadas a otros algoritmos, éstos quedarán traducidos como subprogramas (como funciones cuando son invocados desde una expresión, y como procedimientos cuando el llamado constituye una estructura de control).

Los datos de entrada y de salida del algoritmo principal constituirán los argumentos de los procedimientos predefinidos read o readln y write o writeln respectivamente. Los datos de entrada y los de salida de los algoritmos invocados quedarán traducidos como parámetros. Las variables auxiliares usadas en un algoritmo, quedarán traducidas como variables locales al correspondiente procedimiento (o programa). Dado que la conexión entre algoritmos sólo se hace a través de los datos de entrada y salida, los procedimientos obtenidos en la traducción nunca contarán con variables globales.

2.2.3 El constructor de trazas

Al activarse el constructor de trazas, se solicitará el nombre del algoritmo principal al cual se le asociará la traza. Luego de ingresar los valores de los datos de entrada, se desplegará en pantalla la sucesión de estados de la correspondiente computación. Cuando en el transcurso del desarrollo de una traza, se llega a una invocación de algoritmo, se activará automáticamente el constructor de trazas (sin solicitar el nombre del algoritmo ni los valores para los datos de entrada), desplegándose la correspondiente traza sin que se pierda la anterior a la cual se retornará una vez finalizada la segunda.

3. Conclusiones

El entorno propuesto constituye un recurso valioso para la enseñanza de la programación a los principiantes. Fundamentalmente permite canalizar los esfuerzos hacia la resolución del problema y el desarrollo de los algoritmos. El conocimiento del lenguaje algorítmico como el de programación, podrá ser adquirido naturalmente por los estudiantes en el período de uso del entorno.

De esta manera cuando comienza la etapa de aprendizaje del lenguaje de programación, los estudiantes se encuentran ya familiarizados con muchos aspectos del mismo, a partir de las observaciones de los programas generados por el traductor y de las comparaciones de éstos con los algoritmos correspondientes. Es de esperar, entonces una reducción notable en el tiempo de esta etapa.

La implementación del entorno descripto obviamente se ve favorecida con el empleo de interfaces gráficas. Constituye una aplicación interesante que pueden desarrollar alumnos de grado de Ciencias de la Computación como trabajo final de carrera.

4. Referencias

[Chavey] Darah Chavey- Beloit College. A Structured Laboratory Component for the Introductory Programming Course. SIGCSE BULLETIN ACM. 1991

[Geitz] Robert Geitz. Concepts in the classroom, programming in the lab. ACM SIGCSE BULLETIN. 1994.

[Levy] Lisa Levy Kortright. From Specific Problem Instances to Algorithms in the Introductory Course. . SIGCSE BULLETIN ACM. 1994.

[Peyrin] Jean Pierre Peyrin-. Schemas algorithmiques fondamentaux. Masson. 1989

[Pyott] Pyott, S. and Sanders,I. 'ALEX: an aid too teaching algorithms'. ACM SIGCSE BULLETIN. 1991.

[Terry] Terry, P. 'Umbriel - Imperative Programming for unsophisticated students'. ACM SIGCSE BULLETIN. 1995.

[Wu] Wu C.Thomas. Teaching OOP to beginners. Journal of Object-Oiented Programming. 1993.

[Rueda] Rueda, S. ; Castro, S y Zanconi, M. Resolución de Problemas y Algoritmos: notas de curso. 1994.